#10
1-13-04
DW

In the United States Patent and Trademark Office

Serial number 09/340,172
Application filed June 25, 1999
Applicant Derek Wong
Application Title Methods for Increasing Instruction-Level Parallelism in Microprocessors and Digital Systems
Examiner/GAU Eric Coleman / 2183

Mailed December 22, 2003
at San Jose, California

From:
Derek Wong
1341 Echo Valley Dr.
San Jose, CA 95120

To:
Assistant Commissioner for Patents
United States Patent and Trademark Office
Washington, DC 20231

RE: Information Disclosure Statement for patent application 09/340,172

Thank you for reviewing patent application 09/340,172.

The applicant encloses this information disclosure statement consisting of 15 references. An IDS filing fee of $180 is enclosed.

1. References 1-5 are for background information. These references 1-5 present aspects of conventional superscalar microprocessor design. These are all the cited references from phase I reports from a PCT application PCT/US99/14299 filed independently. The present claims in this application 09/340,172 have evolved greatly over time and are substantially more limited and better stated compared to the original claims in the PCT application. The PCT search references do not appear to be highly relevant to the present

claims in this application 09/340,172, but the applicant is citing the references anyway to be conservative. None of these references 1-5 appear to be close to the presently claimed inventions in this patent application.

2. References 6 through 8 present methods of instruction packing for more efficient execution.
3. Reference 6 is a long reference - a Ph.D. dissertation by John Johnson. It is physically in two paper bundles (part 1 and 2) due to the size. This dissertation describes a technique called Expanded Parallel Instruction Cache (EPIC). Description of the operation of EPIC is described starting on page 20. See Figure 2.1 on page 21 for a basic diagram.
    a. In EPIC, instructions are read from Instruction Memory and packed into VLIW-like instruction storage in the Expanded Instruction Cache (see Figure 2.1). This packing process is called the "expansion process". It puts several instructions that can be executed at the same time into one VLIW-like instruction cache entry.
    b. The expansion process puts instructions in order from the Instruction Memory into instruction cache entries. See section 2.2.1 starting on page 24. The process of packing for a particular cache entry ends when an instruction is reached that would have a dependency on another instruction already in that cache entry or if a required resource in the execution unit is already in use by instructions already in that cache entry. For example, if instructions 1, 2, and 3 are not dependent on each other and don't conflict in execution resources, then they can be placed into the same instruction cache entry. No instruction re-ordering is performed as the expansion process just chooses instructions in order that can be independently executed.

c. The expansion process can accommodate branches by packing instructions from the predicted branch target (for a conditional branch) or unconditional branch target into the existing instruction cache entry. So the expansion process can help reduce the cache fetches caused by branches.

d. Compared to this patent application's claimed invention, Figure 2.1 of Johnson shows that the execution system can only operate on instructions from the Expanded Instruction Cache. There is no cache structure to store instructions without performing the EPIC expansion process. Since there is no way for the execution unit to execute instructions that are not stored in the Expanded Instruction Cache, the execution unit is forced to wait for the expansion process if the Expanded Instruction Cache misses.

e. Also, Johnson's expansion unit is much more limited than this patent application's instruction stream transformation unit. Johnson describes an in-order packing process only. EPIC does not perform any instruction re-ordering, which makes it a much simpler and more limited design. Furthermore, EPIC does not support predicated execution or predictions using any of several tables proposed in this patent application.

f. Johnson's Expanded Instruction Cache stores VLIW-like words. The size is limited to the VLIW width. There is no mechanism for chaining together instruction sequences that are longer than one word. Note that the successor field in the cache entry shown in Figure 2.7 of Johnson is a branch prediction address as explained on pages 28-29 of Johnson. The successor field is not

equivalent to a data chaining structure like the hyperblock ID's and line numbers/pointers in the present patent application.

g. Info on sections in Johnson:

   i. Section 2.3 describes a conventional superscalar execution model called Reorder Buffer Model. This conventional superscalar machine does not use the EPIC design. This Reorder Buffer Model is used as a baseline reference in later performance model comparisons.

   ii. Section 2.4 (Figure 2.15) describes a machine with an EPIC unit connected to an out-of-order execution unit. This is called Expro. In this case, the expansion process is modified so that instructions that are dependent or that have a resource conflict can be packed into the same expanded instruction cache entry. The out-of-order execution unit will manage these dependencies during execution. So the main function of the EPIC unit in Expro is to consolidate instructions and reduce cache fetches caused by branches. Note the expansion process in Expro still packs instructions in order and does not perform any re-ordering. Only the execution unit can execute out-of-order.

   iii. Chapter 3 (starting on page 69) describes the compiler scheduling requirements for EPIC machines. This is a discussion of how a compiler should order instructions so that compiled programs can be executed efficiently by EPIC-type designs. This chapter is not a description of what the EPIC unit itself does. A list scheduling algorithm for the compiler is described. See pages 82-83 for a summary.

iv. Chapters 5-8 describe more details of how to do instruction packing and branch prediction.

4. Reference 7 by Rotenberg et al describes an instruction packing mechanism called a trace cache. A trace cache holds "execution traces" which consist of sequences of instructions in the order that they are executed at run-time. This is similar to the packing concept in Johnson's Expanded Parallel Instruction Cache. Rotenberg's trace cache stores the instructions in fetch order into cache lines; instructions are not reordered or rescheduled.

   a. *This reference's design is limited to handling traces that are no larger than a single cache line.* The cache line size of *n instructions* sets the maximum length of a trace (see section 1.1 of reference 7). No mechanism is proposed for transforming nor storing traces longer than one cache line.

   b. Thus Rotenberg's design is much more limited than this patent application's instruction stream transformation unit and instruction stream cache.

      i. The transformations in reference 7 are limited to instruction packing. The main goal of reference 7 is to enable the packing of instructions from several related basic blocks into one trace cache line so they can be fetched in one cache access, rather than fetching non-contiguous basic blocks from separate addresses which likely results in multiple cache accesses.

      ii. Unlike the instruction stream transformations in the present patent application, the trace cache in reference 7 has no instruction re-ordering, no predication, or other operations.

iii. Traces in reference 7 are limited in to one cache line in length. In contrast, the instruction stream cache and transformation unit in the present patent application can handle hyperblocks that are much longer.

5. In reference 8, Rotenberg et al describes a trace processor. A trace processor is shown in Figure 1 of reference 8. This trace processor contains an instruction cache that feeds a trace construct unit that feeds trace preprocess and then the trace cache. The execution units, called processing elements in Figure 1, can only execute results from the trace cache and cannot directly execute instructions from the instruction cache.

   a. This is a proposed processor design based on the trace cache of reference 7. The idea is to first create and store traces and then use processing elements (PE's) that take instruction input at the granularity of traces rather than individual instructions.

   b. The trace cache in reference 8 is similar in concept to the one in reference 7.

The present patent application's design can transform instruction sequences that are much longer than the instruction packing done in either Johnson's or Rotenberg et al's design. An important feature in the present patent application is therefore the means to store these longer instruction sequences in the instruction stream cache as a group of cache lines. This storage is accomplished using data structures such as hyperblock ID tags and line numbers or pointers to chain together multiple cache lines into one hyperblock, as described on pages 19-22 of the patent application. Johnson and Rotenberg et al do not have an equivalent mechanism.

7. References 9-11 contain background information regarding the Intel IA-64 / EPIC design. (Note: Intel's EPIC stands for Explicitly Parallel Instruction Computing. This

is not the same abbreviation as Johnson's Expanded Parallel Instruction Cache in reference 6, also abbreviated as EPIC).

a. Reference 9 by Crawford and Huck is a set of presentation slides with speaker notes describing the IA-64 / EPIC architecture.

b. Reference 10 by Crawford and Huck is a transcript of the presentation.

c. Reference 11 is a Web page from intel.com.

d. These files were downloaded individually from the intel.com Web site. The applicant did not attend the conference where the paper was presented and doesn't have the corresponding proceedings.

8. References 12-15 contain background information.

a. Reference 12 by Gallagher et al describes a method of dynamic memory disambiguation using a memory conflict buffer. This is a different concept than a history table as described in this patent application.

b. Reference 13 by Wang and Franklin describes a method of using a value history table to predict data values. This is shown in Figure 1 of the reference.

c. Reference 14 by August et al and reference 15 by Mahlke et al describe methods of predicated execution and if-conversion in compilers. These papers are about compilers rather than processor designs. Neither of these references describes anything like an instruction stream transformation unit coupled to an instruction stream cache in a hardware processor. Neither do these references show the concept of a predicate history table, which is contained in the present application.

d. These references were downloaded as individual files from the Internet. The applicant did not attend the conferences and does not have the corresponding conference proceedings for those papers which were presented at conferences.

The applicant is available at telephone number 408-927-7940 to discuss this application.

Thank you for your review of this application and thank you for your help.

Very respectfully,

Derek Wong

Derek Wong

Patent Applicant

Express Mail Label # EU727223990US Date of Deposit December 22, 2003

I hereby certify that this paper or fee is being deposited with the United States Postal Service using "Express Mail Post Office to Addressee" service under 37 CFR 1.10 on the date indicated above and is addressed to "Assistant Commissioner for Patents, Washington, DC 20231."

Signed Derek Wong Inventor